
Salud

Contenido

MANUAL DE CONFIGURACIÓN SO-DEBIAN Y OTROS AJUSTES DE ARQUITECTURA

Instalación de Postgres12 y pgAdmin4:

sudo apt update

sudo apt-get install curl ca-certificates gnupg

curl https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -

echo "deb http://apt.postgresql.org/pub/repos/apt/ 'lsb_release -cs'-pgdg main" | sudo tee /etc/apt/sources.list.d/pgdg.list

sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'

Una vez hecho esto, tenemos que actualizar las cabeceras

sudo apt update
sudo apt -y install postgresql-12 postgresql-client-12

Instalamos pdAdmin4
https://geeksencuarentena.com/linux/como-instalar-postgresql-y-pgadmin4-en-ubuntu-20-04/

curl https://www.pgadmin.org/static/packages_pgadmin_org.pub | sudo apt-key add
$ sudo sh -c 'echo "deb https://ftp.postgresql.org/pub/pgadmin/pgadmin4/apt/$(lsb_release -cs) pgadmin4 main" > /etc/apt/sources.list.d/pgadmin4.list && apt update'

Luego instalar **pgAdmin4**,

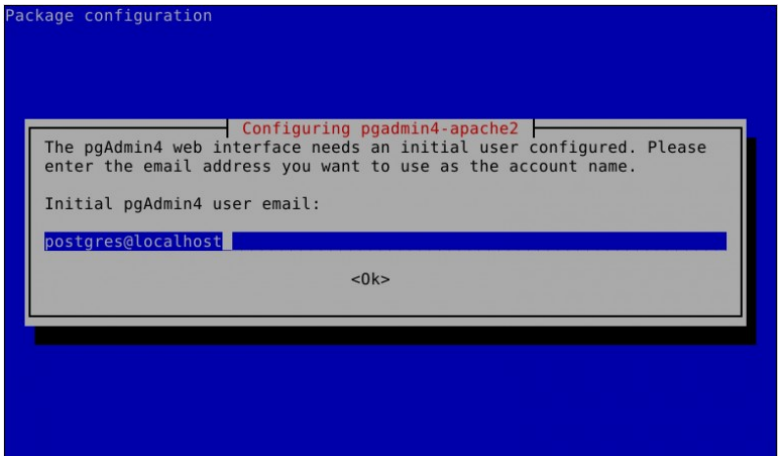
$sudo apt install pgadmin4

sudo /usr/pgadmin4/bin/setup-web.sh

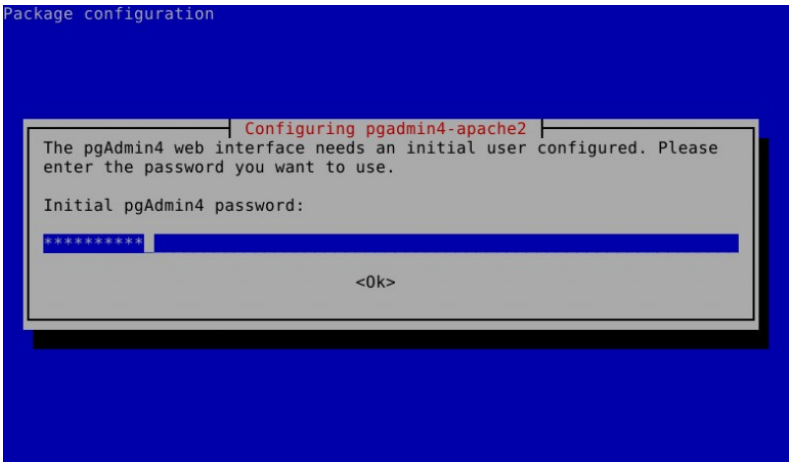
correo: postgres@localhost
password: 123456

```
root@tecmint1:~# apt-get install pgadmin4 pgadmin4-apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  alembic apache2 apache2-bin apache2-data apache2-utils fonts-open-sans
  fonts-roboto-unhinted libapache2-mod-wsgi-py3 libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0 pgadmin4-common
  pgadmin4-doc python-babel-localedata python3-alembic python3-asn1crypto
  python3-babel python3-bcrypt python3-blinker python3-cffi-backend
  python3-click python3-colorama python3-crypto python3-cryptography
  python3-dateutil python3-distutils python3-editor python3-flask
  python3-flask-babelex python3-flask-gravatar python3-flask-htmlmin
  python3-flask-login python3-flask-mail python3-flask-migrate
  python3-flask-paranoid python3-flask-principal python3-flask-security
  python3-flask-sqlalchemy python3-flaskext.wtf python3-htmlmin
  python3-itsdangerous python3-jinja2 python3-lib2to3 python3-mako
  python3-markupsafe python3-nacl python3-openssl python3-paramiko
  python3-passlib python3-psutil python3-psycopg2 python3-pyasn1
  python3-simplejson python3-sqlalchemy python3-sqlalchemy-ext
  python3-sqlparse python3-sshtunnel python3-tz python3-werkzeug
  python3-wtforms
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
```

Durante la instalación se nos preguntara la dirección de correo del usuario, los dejamos con la dirección que tiene como defecto y damos en “Ok”.



Después se pedirá que ingrese la contraseña que desea usar con pgAdmin4



Una vez que los paquetes están instalados. El instalador inicia y habilita el servicio Apache2, para revisar el estado del servicio, escribe el siguiente comando

systemctl status apache2

Antes de iniciar pgAdmin4 tenemos que habilitar los puertos. Para poder hacerlo necesitamos tener instalado el siguiente programa.

sudo apt-get install ufw

Después para abrir los puertos necesarios:

sudo ufw allow 22
sudo ufw allow 80
sudo ufw allow 443
sudo ufw allow 8080
sudo ufw allow 8443
sudo ufw allow 57361
sudo ufw enable
sudo ufw status

(aveces es necesario agregar la palabra sudo antes de la instrucción para que se ejecute)

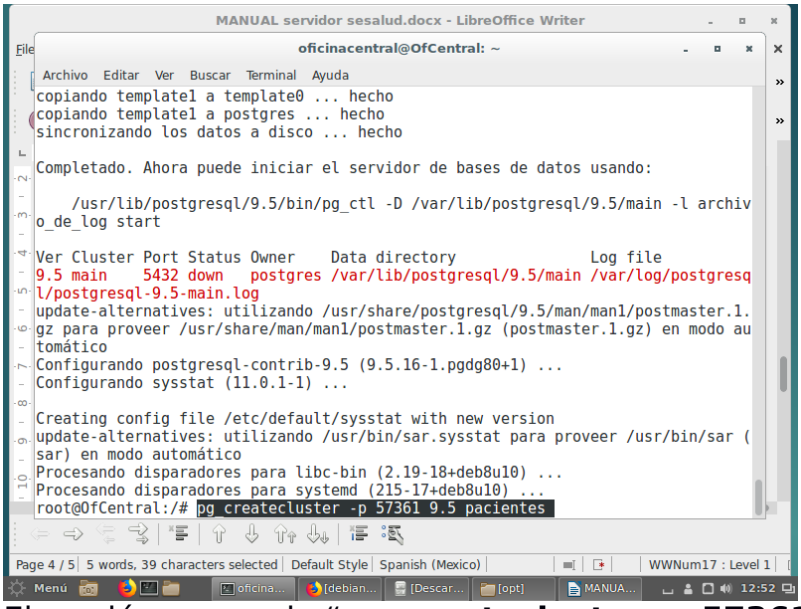
Y con esto ya puedes abrir pgAdmin4, cuando lo abras pedirá que introduzcas una contraseña nueva para usar con el usuario de pgAdmin4, una vez que la introduzcas ya podrás conectarte a los servidores y manipular las bases de datos

Configuración Postgresql:

En el manual 1 se indicó el procedimiento a seguir para la instalación de Postgresql (según la versión que se instaló en el manual 1) y aquí se detallará el proceso a seguir para su configuración.

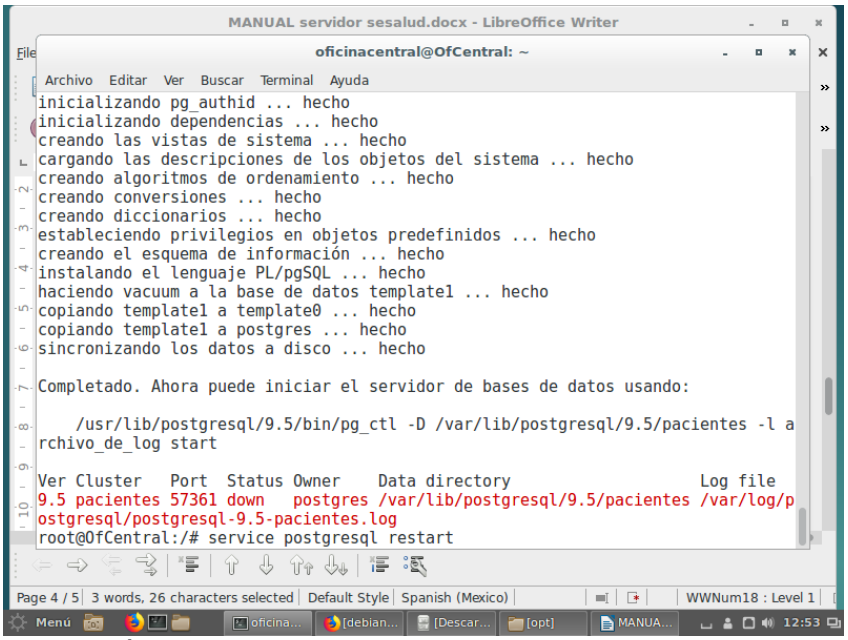
Iniciamos creando el nuevo cluster que almacenará nuestra base de datos de pacientes, lo hacemos mediante la siguiente instrucción:

- # sudo **pg_createcluster -p 57361 12 pacientes**



Ejecución comando "**pg_createcluster -p 57361 12 pacientes**"
Una vez creado, deberemos iniciar los servicios de PostgreSQL:

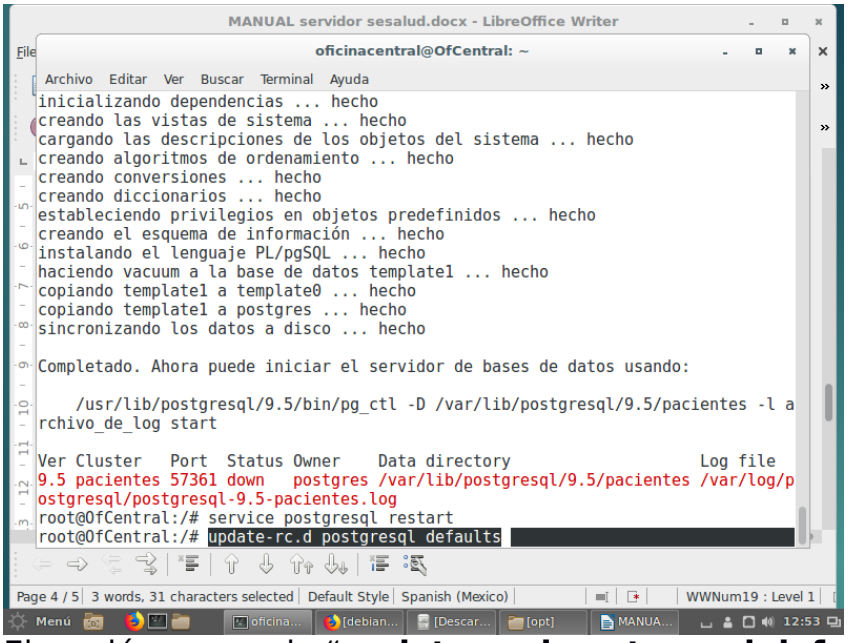
- **# sudo service postgresql restart**



Ejecución comando "**service postgresql restart**"

Así mismo queremos que PostgreSQL se inicie de forma automática junto con el sistema, lo hacemos de la siguiente forma:

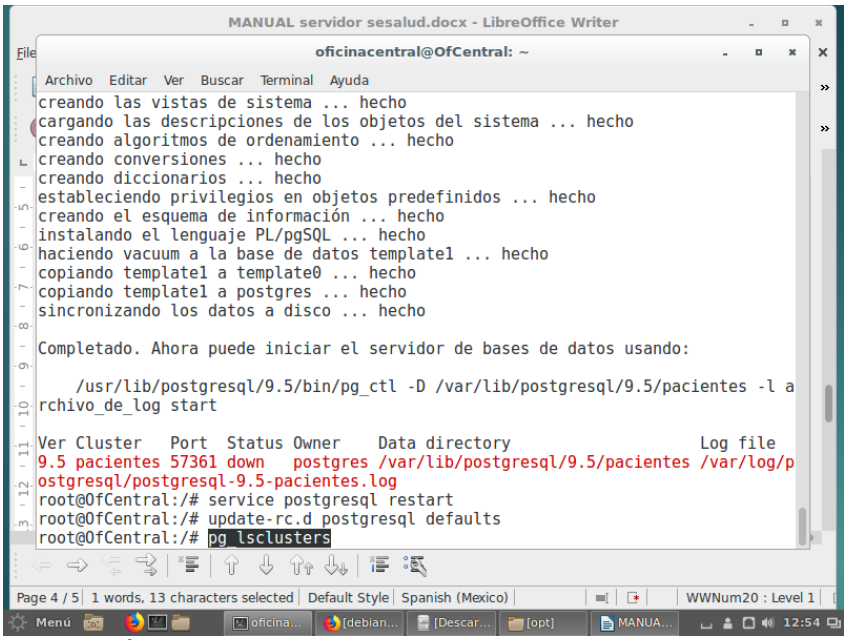
- **# sudo update-rc.d postgresql defaults**



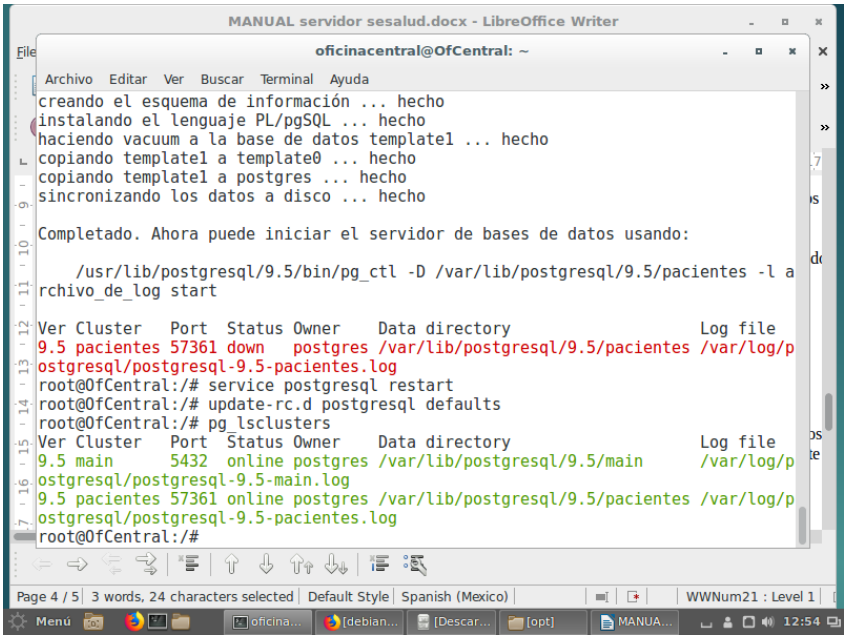
Ejecución comando "**update-rc.d postgresql defaults**"

Finalmente podremos comprobar el correcto funcionamiento de los clusters con el siguiente comando:

- **# sudo pg_lsclusters**



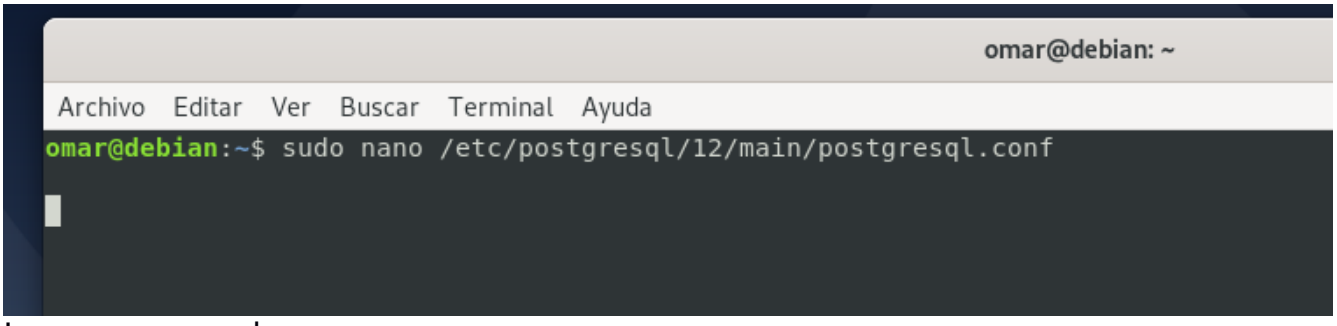
Ejecución comando "**pg_lsclusters**"



Y obtendremos un resultado en letras verdes indicando el status **online.**

Primero se deben modificar los archivos de configuración que más adelante permitirán a otras computadoras ingresar al servidor y su Base de datos.
Para realizar tal acción deberemos ingresar mediante consola en modo root y editar los siguientes archivos:

root@debian:~# sudo nano /etc/postgresql/12/pacientes/postgresql.conf



Ingreso comando

Nos desplazamos a la sección **Connections and authentication** y modificamos la línea que contiene **listen_addresses**, quitando en símbolo # al inicio y quedando de la siguiente forma:

```
#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -

listen_addresses = '*'          # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost'; use '*' for all
                                # (change requires restart)
port = 57361                    # (change requires restart)
max_connections = 50            # (change requires restart) 100
# Note:  Increasing max_connections costs ~400 bytes of shared memory per
# connection slot, plus lock space (see max_locks_per_transaction).
#superuser_reserved_connections = 3     # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of directories
                                # (change requires restart)
#unix_socket_group = ''                 # (change requires restart)
#unix_socket_permissions = 0777         # begin with 0 to use octal notation
```

Verificar que la siguiente linea : **max_locks_per_transaction=64**

```
#------------------------------------------------------------------------------
# LOCK MANAGEMENT
#------------------------------------------------------------------------------

#deadlock_timeout = 1s
max_locks_per_transaction = 64          # min 10
                                        # (change requires restart)
# Note:  Each lock table slot uses ~270 bytes of shared memory, and there are
# max_locks_per_transaction * (max_connections + max_prepared_transactions)
# lock table slots.
#max_pred_locks_per_transaction = 64    # min 10
                                        # (change requires restart)


#------------------------------------------------------------------------------
# VERSION/PLATFORM COMPATIBILITY
#------------------------------------------------------------------------------

# - Previous PostgreSQL Versions -
```

Verificar que la siguiente linea tenga el formato: “dmy”

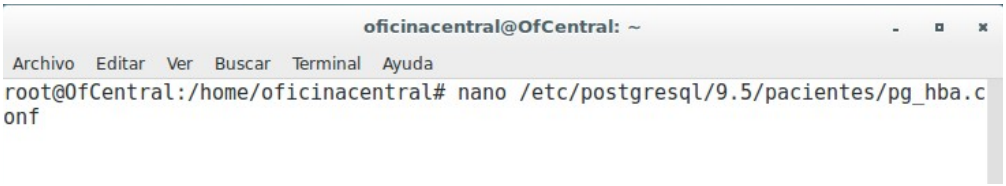
```
#xmlbinary = 'base64'
#xmloption = 'content'
#gin_fuzzy_search_limit = 0

# - Locale and Formatting -

datestyle = 'iso, dmy'
#intervalstyle = 'postgres'
timezone = 'localtime'
#timezone_abbreviations = 'Default'     # Select the set of available time zo
                                        # abbreviations.  Currently, there ar
                                        #   Default
                                        #   Australia (historical usage)
                                        #   India
                                        # You can create your own file in
                                        # share/timezonesets/.
#extra_float_digits = 0                 # min -15, max 3
#client_encoding = sql_ascii            # actually, defaults to database
```

Guardamos cambios en el editor nano (CTRL + o) y salimos del editor nano (CTRL + x).

Editando archivo postgresql

Ahora procedemos a editar el siguiente archivo **pg_hba.conf** de la siguiente manera:

root@debian:~# sudo nano /etc/postgresql/12/pacientes/pg_hba.conf



Editando archivo pg_hba.conf

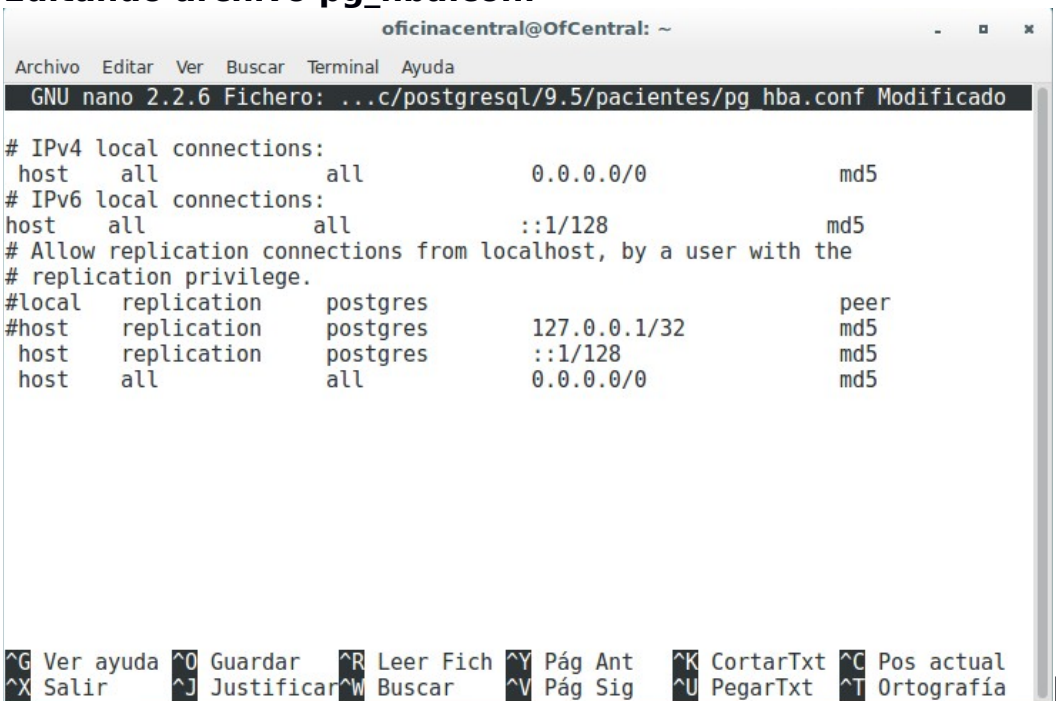
Nos desplazamos al final del archivo y quitamos el símbolo # que está al inicio de la última línea, además agregamos la siguiente línea de texto al final del archivo, los espacios entre palabras son tabulaciones:

host all all 0.0.0.0/0 md5

Quedando las dos últimas líneas de la siguiente forma:

host replication postgres ::1/128 md5
host all all 0.0.0.0/0 md5

Editando archivo pg_hba.conf



host

Guardamos cambios en el editor nano (CTRL + o) y salimos del editor nano (CTRL + x).

Procedemos a reiniciar el servicio de base de datos postgres para que los cambios tomen efecto:

Reiniciando servicio postgresql

$sudo /etc/init.d/postgresql stop

y luego lo iniciamos con

$sudo /etc/init.d/postgresql start

```
root@sesalud:~# /etc/init.d/postgresql start
Starting postgresql (via systemctl): postgresql.service.
```

Una vez que se reinicie, procederemos a cambiar la contraseña del usuario postgres para que así se pueda acceder a la base de datos que crearemos más adelante.

sudo -i
sudo passwd root
admin20//
Entramos a postgres con el siguiente comando.

$su postgres

Ahora cambiaremos la contraseña del usuario postgres.

$psql -U postgres -p 57361 -c "ALTER USER postgres WITH PASSWORD '12345'; "

```
exit
root@serverCharcas:~# su - postgres
postgres@serverCharcas:~$ psql -U postgres -p 57361 -c "ALTER USER postgres WITH PASSWORD '12345';"
ALTER ROLE
postgres@serverCharcas:~$
```

Se agregará un usuario sin trigger para que la base de datos no valide con las reglas de control (triggers)

$psql -p 57361 -c "CREATE ROLE sin_triggers_ WITH LOGIN SUPERUSER PASSWORD '12345';"

```
postgres@sesalud:/root$ psql -p 57361 -c "CREATE ROLE sin_triggers_ WITH  LOGIN  SUPERUSER PASSWORD '12345';"
could not change directory to "/root": Permiso denegado
CREATE ROLE
```

Se crea la BD seg_pac

$psql -p 57361 -c "CREATE DATABASE seg_pac;"

```
postgres@sesalud:/root$ psql -p 57361 -c "CREATE DATABASE seg_pac"
could not change directory to "/root": Permiso denegado
CREATE DATABASE
```

Restaurar la base de datos vacía:

https://drive.google.com/file/d/1GqvCFJKykI3JXhMZZROTK4hy_N9B-1DQ/view?usp=sharing

Realizar la importación primero debemos ubicar el archivo sesalud_vacio_*[version]*.7z ubicado en la carpeta de GDrive AplicaciónYmdb, descargarlo y descomprimirlo en una ubicación conocida (por ejemplo: /home/*[server]*/Descargas/sesalud_vacio_*[versión]*.sql).

Finalmente procedemos a escribir el siguiente comando en la consola para iniciar la importación:

$psql -h localhost -U sin_triggers_ -p 57361 seg_pac -f /ruta_donde_este_guardado_la_base_de_datos/sesalud_vacia.sql

A lo que solicitará la contraseña para el usuario sin_triggers_ escribimos “12345” (sin comillas) y damos enter.

Se comenzarán a importar todos los datos del contenido en el archivo y debemos esperar a que nos vuelva a mostrar el carácter $ para saber que ha concluido:

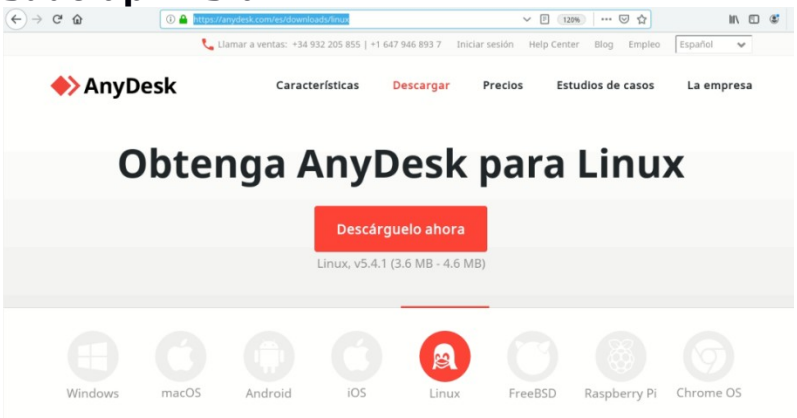
Instalar aplicación AnyDesk:

Descargar de la página oficial :
https://anydesk.com/es/downloads/linux

wget https://download.anydesk.com/linux/anydesk_6.1.0-1_amd64.deb

sudo dpkg -i anydesk_6.1.0-1_amd64.deb

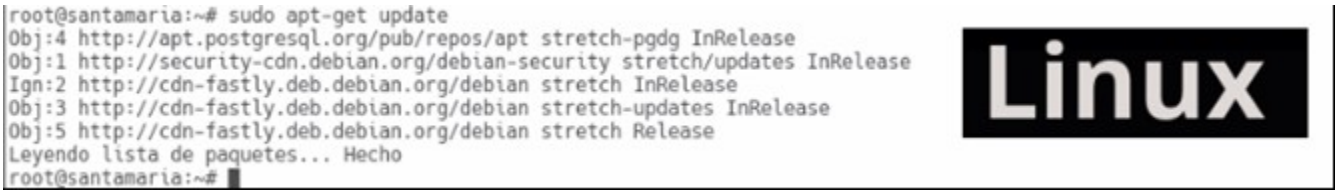
sudo apt install -f



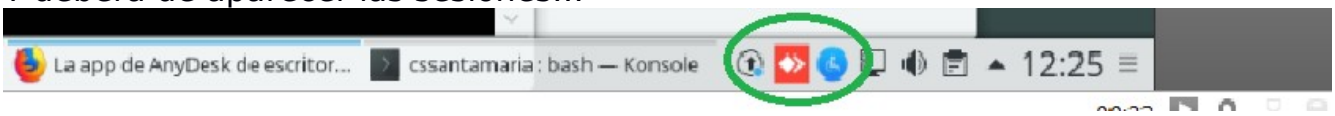
Se descarga extensión ".deb"

Ubicar el archivo y acceder a la carpeta donde se encuentra "cd /home/[unidad]/Descargas/"
Instalador de paquetes

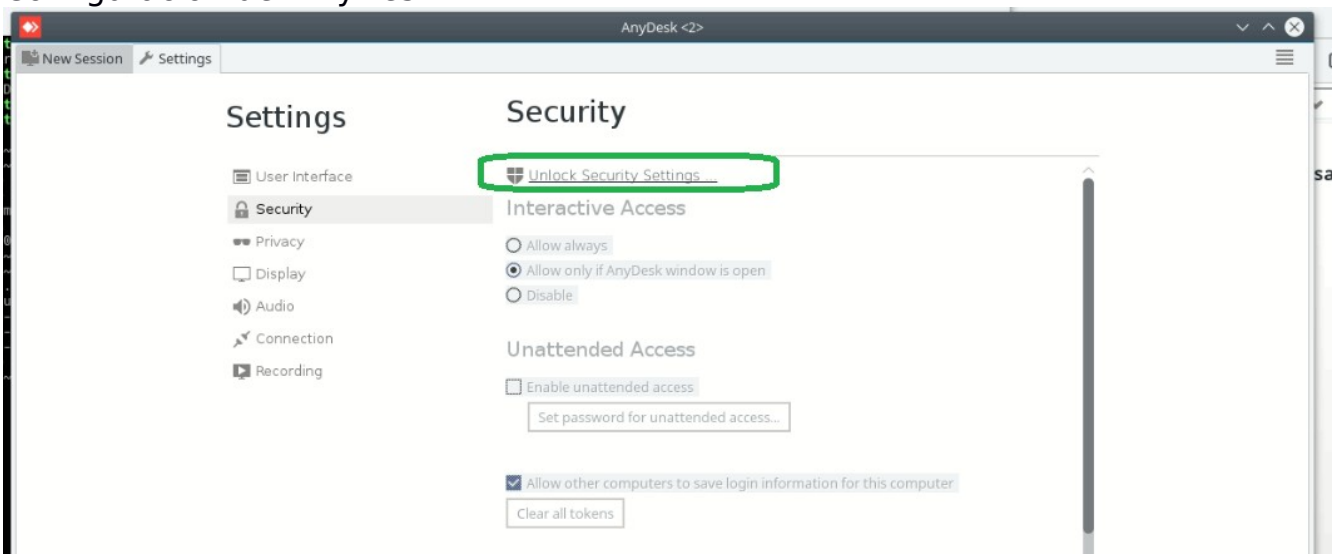
sudo dpkg -i anydesk_5.4.1-1_amd64.deb



Y deberá de aparecer las sesiones...



Configuración del AnyDesk



Aplicar la contraseña para realizar conexión remota.

Instalar Tomcat

Para instalar tomcat necesitamos tener el sistema actualizado, por lo que sería recomendable ejecutar los comandos **sudo apt-get update** seguido de **sudo apt-get upgrade.**

Instalando Java

Necesitamos el entorno de ejecución, se instala con el siguiente comando:

sudo apt install -y default-jre

Ahora instalaremos el JDK que es el soporte para desarrollo
Nos movemos a la ubicación siguiente.

/usr/local/src/

Después los descargamos por medio del siguiente comando:

wget http://download.oracle.com/otn-pub/java/jdk/6u25-b06/jdk-6u25-linux-i586-rpm.bin -O jdk-6u25-linux-i586-rpm.bin

Después ejecutamos el siguiente comando

bash jdk-6u25-linux-i586-rpm.bin

Vamos a añadir las siguientes variables de ambiente de java, entramos al siguiente archivo para agregarlas al final "**/etc/profile**".

```
export JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
```

Note que la variable "JAVA_HOME" puede cambiar según se haya instalado java en nuestro equipo, para asegurar que la ruta sea correcta, podemos buscar la ruta de instalación de java hasta que encontremos la carpeta "bin", así tendríamos la ruta completa de "JAVA_HOME"

Y ahora ejecutamos el perfil para que se cargue.

```
. /etc/profile
```

Vamos a crear la variable también en el siguiente archivo **/etc/environment,** para abrirlo usamos **nano /etc/environment** y escribiremos lo siguiente:

```
JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
```

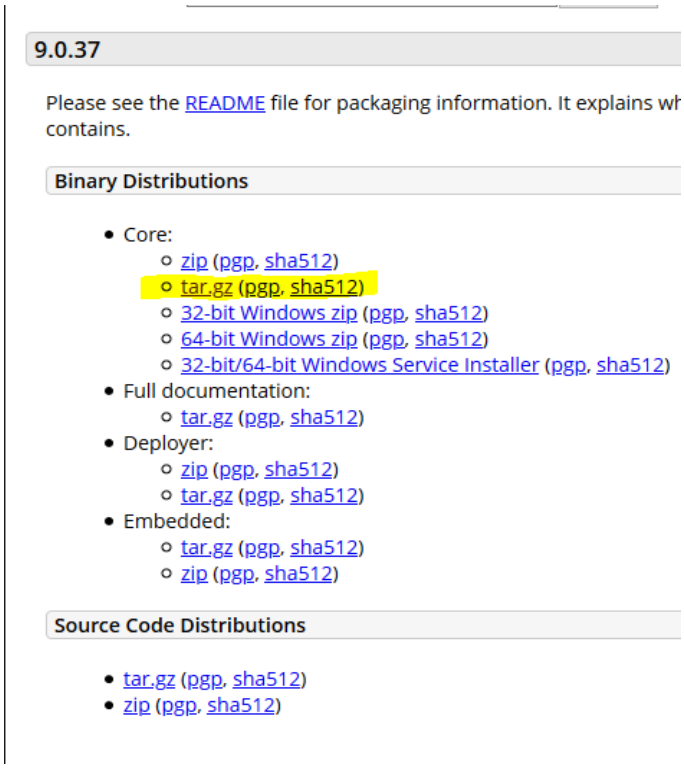
Guardamos y ejecutamos el comando **source /etc/enviroment,** para comprobar que la variable está establecida ejecutamos **echo $JAVA_HOME** y nos debería de aparecer la ruta que le hemos dado como valor.

Instalando Tomcat

Actualizamos del sistema con sus comandos correspondientes

sudo apt-get update
sudo apt-get upgrade

Nos ubicamos en **/usr/local/src/** para poder descargar el .tar de tomcat que deseamos.



Iremos a la página de descargas de tomcat, en https://tomcat.apache.org/download-90.cgi

Una vez ubicados ahí, procedemos a copiar la dirección del enlace del archivo tar, de la versión de tomcat que deseamos instalar.

Usaremos el siguiente comando para poder descargar el archivo en la ubicación donde estamos actualmente.

wget https://downloads.apache.org/tomcat/tomcat-9/v9.0.43/bin/apache-tomcat-9.0.43.tar.gz

a continuación, descomprimimos el archivo

tar xzf apache-tomcat-9.0.37.tar.gz -C /opt

Ahora crearemos un enlace desde tomcat a la carpeta donde ira instalaran todos los archivos.

ln -s /opt/apache-tomcat-9.0.37/ /opt/tomcat

añadimos el usuario tomcat que será el propietario de las carpetas de instalación

de tomcat

useradd tomcat

Ahora lo hacemos propietario de los directorios

chown tomcat. /opt/tomcat/ -R

Ya podemos iniciar el servicio, nos posicionamos en **/opt/tomcat/bin** y procedemos a iniciarlo con los siguientes comandos:

./startup.sh (encender) **./shutdown.sh (apagar)**

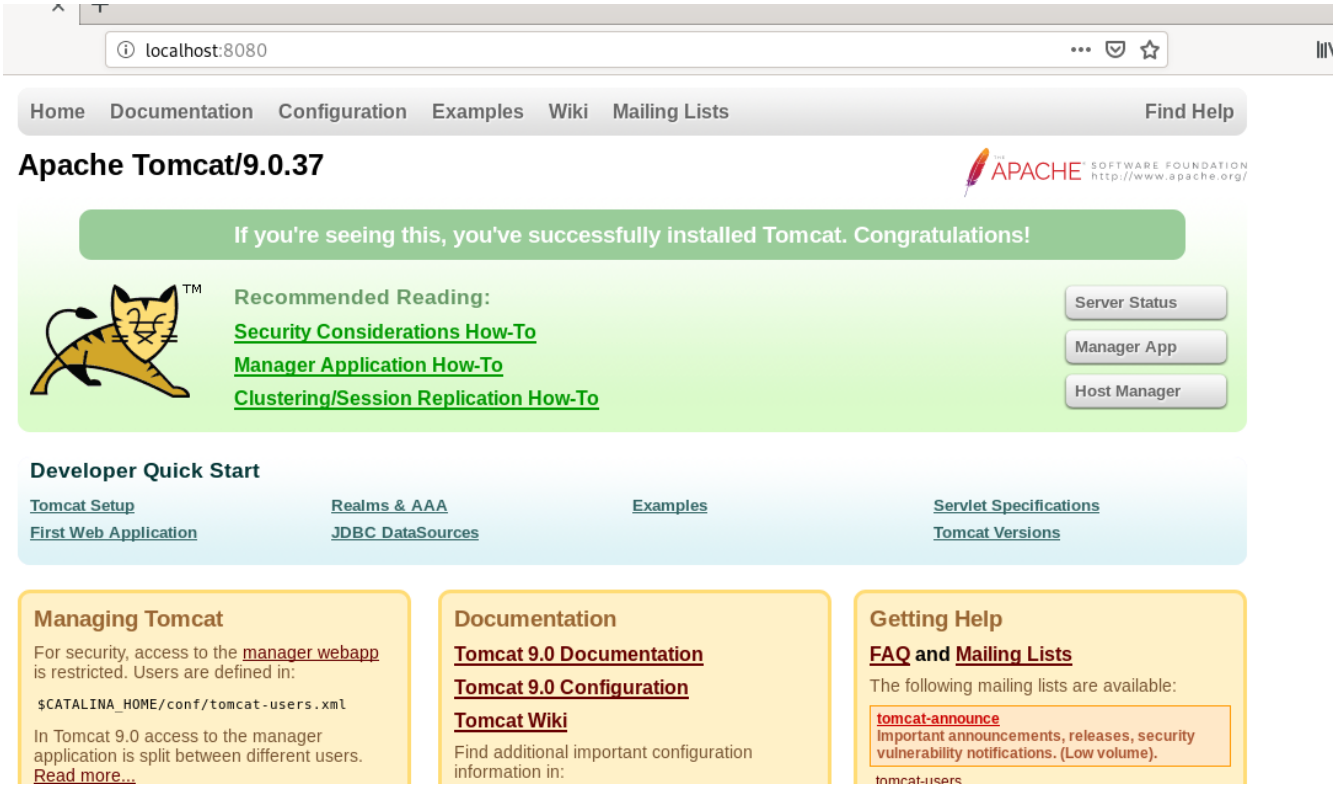
o bien

./catalina.sh start (encender) **./catalina.sh (stop)**

Si todo está correcto no mostraría la siguiente información:

```
$ ./startup.sh
Using CATALINA_BASE:   /opt/tomcat
Using CATALINA_HOME:   /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME:        /usr/lib/jvm/java-1.11.0-openjdk-amd64
Using CLASSPATH:       /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-
juli.jar
```

Podemos probar el servidor escribiendo en cualquier navegador *http://localhost:8080*



Nota: si por algún motivo el servidor no puede acceder a tomcat, ya sea por algún reinicio no programado o por algún apagón, puede probar la siguiente solución para arrancar el servicio.

Ir a la ubicación **/usr/local/src/** y verificar por medio del comando **ls** que el archivo llamado "**jdk-6u25-linux-i586-rpm.bin**" se encuentre ahí. Para posteriormente ejecutar el siguiente comando.

bash jdk-6u25-linux-i586-rpm.bin

y al terminar volveremos a ejecutar el archivo profile con:

. **/etc/profile**

Y arrancamos el tomcat con:

sudo sh /opt/tomcat/bin/startup.sh

Esto reconfigurara la variable de java y así poder arrancar el servicio.

-

Configurar usuario administrador en tomcat

Una vez que hemos añadido el usuario tomcat en los pasos anteriores editamos el archivo “tomcat-user.xml” mediante la siguiente ruta:

sudo nano /opt/tomcat/conf/tomcat-users.xml

en este archivo encontraremos un bloque donde están los roles del usuario y su password,
aquí se pueden asignar los usuarios que se necesiten con sus respectivas contraseñas.

```
        <<role rolename="admin-gui"/>
        <role rolename="manager-gui"/>
<user username="tomcat" password="12345" roles="admin-gui,manager-gui"/>
```

Una vez asignados los usuarios, podrá entrar a la pantalla de las aplicaciones que es la siguiente

Permitir la conexión de cualquier host a la manager-app

Ir a las siguientes rutas para editar el archivo context.xml

nano /opt/tomcat/webapps/host-manager/META-INF/context.xml

nano /opt/tomcat/webapps/manager/META-INF/context.xml

nano /opt/tomcat/webapps/host-manager/manager.xml

En la parte donde está el siguiente bloque, “antiResourceLocking” si está en true lo ponemos en false y borramos todas las líneas que estén entre <Context> y </Context> y así debería de quedar

```
<Context antiResourceLocking="false" privileged="true" >
```

```
</Context>
```

Reiniciar tomcat

Modificando la capacidad en Tomcat

Para poder subir un archivo war en tomcat con un tamaño superior a los 50 mb, es necesario configurar algunas líneas de unos archivos de configuración de tomcat.

Iniciamos sesión como root con el comando **su**

Editaremos el archivo "**server.xml**" añadiendo el atributo maxPostSize="104857600" en la sección del conector. Esta en la ruta
nano /opt/tomcat/conf/server.xml

```
<Connector port="8080" protocol="HTTP/1.1"
  connectionTimeout="20000"
  maxPostSize="104857600"
  redirectPort="8443" />
```

Despues para el archivo "web.xml", nos movemos a la ruta del archivo **/opt/tomcat/webapps/manager/WEB-INF/** y lo editamos con el siguiente comando:

nano /opt/tomcat/webapps/manager/WEB-INF/web.xml

Buscar la sección "<multipart-config>", aquí es donde vamos a modificar la capacidad, dejándolo como esta en el siguiente texto:

```
<multipart-config>
      <!-100MB max  en nueva config ->
      <max-file-size>104857600</max-file-size>
         <max-request-size>104857600</max-request-size>
      <file-size-threshold>0</file-size-threshold>
</multipart-config>
```

Guardamos y reiniciamos tomcat con

```
sh /opt/tomcat/bin/shutdown.sh
sh /opt/tomcat/bin/startup.sh
```

Cron para inicio automático de tomcat

Crearemos un cronjob para el usuario root, con esta tarea se iniciará automáticamente el servicio de tomcat cuando reiniciemos el equipo o se vuelva a encender

Entrar con el usuario root con:

su

Una vez que hayamos entrado, escribimos el comando **crontab -e,** si es la primera vez que ejecutamos este comando con ese usuario, nos pedirá que confirmemos el programa de nuestra preferencia para poder editarlo, el más fácil de usar es nano así que seleccionaríamos la opción 1.

Cuando se abra la pantalla para editar nuestro cron, iremos al final del archivo y escribiremos la siguiente línea:

@reboot sh /opt/tomcat/bin/startup.sh

Guardamos con Ctrl+O y cerramos con Ctrl+X.

Recibiremos una respuesta de que el nuevo cron se instaló satisfactoriamente.

Despliegue de una aplicación .war en Tomcat

Iniciar sesión como root con el comando **su**
Ir a la carpteta de Google Drive y descargar el archivo llamado **sesalud_interfaz_15072020.7z**

https://drive.google.com/file/d/14hnwZncmsoQvvgg5gpZuVK3nuoFupnMG/view?usp=sharing

nos posicionasmos en /home/”nombreservidor”/Descargas
wget https://slp-db.slpsalud.gob.mx:8443/soporte/web/sesalud_file/gui/sesalud_interfaz_05012021.7z

Cuando los descarguen, moverse a la carpeta donde descargo, comúnmente es en Descargas, puede hacerlo en modo grafico o desde la terminal.
Para descomprimir el archivo podemos ejecutar el siguiente comando:

7z x sesalud_interfaz_05012021.7z

Después de descomprimir vamos a copiar el archivo a la carpeta de aplicaciones de tomcat con el siguiente comando

$cp sesalud.war /opt/tomcat/webapps

y listo ahora en el navegador podemos acceder a la aplicación por la siguiente dirección:
localhost:8080/sesalud

Actualizar bases de datos

Nos posicionamos en /home/"nombredeservidor"/Descargas
wget
https://slp-db.slpsalud.gob.mx:8443/soporte/web/sesalud_file/sbd/sesalud_script_1.028.sql

entramos
su postgres
psql -h localhost -U sin_triggers_ -p 57361 seg_pac -f /home/sesalud/Descargas/sesalud_script_1.028.sql

sudo sh /opt/tomcat/bin/shutdown.sh
sudo sh /opt/tomcat/bin/startup.sh

Verificar ruta de Temporales

Tenemos que verificar la ruta de los archivos temporales, una que la aplicación war este desplegada, editamos a el archivo **constantes.properties** con el siguiente comando **nano /opt/tomcat/webapps/sesalud/WEB-INF/classes/constantes.properties,** remplazamos la ruta de la última línea por "**/opt/tomcat/sesalud_temp"**

Respaldo de la base de datos por script

En esta sección se creará un script que respaldara automáticamente la base de datos en el servidor y según sea el caso, si el servidor cuenta con conexión a internet, el respaldo se enviara a el servidor principal, y en cambio si no hay conexión a internet, se guardara en una memoria usb que tendrá que estar conectada en el servidor.

Estableciendo contraseña automática para postgres

Para poder ejecutar el script de forma automática, debemos de crear un archivo donde almacenemos la contraseña del usuario postgres o del usuario que es propietario de la base de datos que queremos respaldar.

Nos ubicaremos en la carpeta de root con **cd /root/** después de eso, crearemos un el archivo con nano de la siguiente manera **nano .pgpass** es importante que lleve el "." ps así establecemos que el archivo será oculto y solo el usuario al que le pertenezca el archivo podrá verlo.

Cuando se abra el editor pondremos la siguiente línea de configuración:

localhost:57361:seg_pac:postgres:12345

Formato de la línea "host:puerto:base de datos:usuario:contraseña"

Guardamos con Ctrl+O y cerramos con Ctrl+X

Ahora le daremos permisos de lectura y escritura con el comando **chmod 600 /root/.pgpass**

Con ello configurado así, el script no pedirá que el usuario teclee la contraseña de postgres

Creando un script en Linux.

Para crear un script que nos ayude a crear el respaldo de la base de datos, primero tenemos que ubicarnos por medio de la terminal en el directorio donde está instalado postgres.

Comúnmente la ruta es la siguiente:

```
cd /var/lib/postgres
```

Cuando estemos en el directorio de postgres, con el comando **ls,** podremos ver que solo se encuentra una carpeta con el nombre de la versión que tenga instalado de postgres, en este directorio.

```
omar@debian:/var/lib/postgresql$ ls
12  cron
```

Script para enviar al servidor principal

Para este script necesitaremos instalar un programa especial que nos permitirá guardar la contraseña del servidor principal , para que el usuario no tenga la necesidad de escribir ninguna contraseña.

Ejecutamos el siguiente comando **sudo apt-get install -y expect**

Después de que este instalado, crearemos un nuevo archivo con “nano” con el siguiente comando **nano enviarRespaldo.exp**

Se abrirá el editor, y deberemos de escribir la siguiente línea.

#!/usr/bin/expect

Esto es la ruta donde se acaba de instalar el programa que bajamos, y utilizara las librerías de esa ubicación.

Después necesitamos establecer una variable que se nos será dada por medio de un parámetro. Con la siguiente línea.

set nombre [lindex $argv 0];

En esta línea, se esta colocando una variable que se llama “nombre” y le damos el valor indicado entre los corchetes.

El siguiente paso es desactivar el temporizador para que el programa termine su ejecución por completo.

set timeout -1

A continuación, colocamos la línea que será el comando que queremos ejecutar, que es enviar el respaldo a el servidor principal en una ruta específica, precedido por el comando “spawn” ya que este será el que enlace el comando con la respuesta que tiene la contraseña

spawn scp $nombre sesalud@slp-db.slpsalud.gob.mx:/home/sesalud/respaldos

ahora colocamos la variable “pass” con su valor

set pass sesalud

Para finalizar el script, llamamos a la función expect que esperara que la terminal pida una contraseña, y una vez que lo haga, enviara la contraseña que hemos definido en el paso anterior, agregándole “\r” para simular un “enter”, y después con la instrucción “exp_continue” el programa continuara la ejecución del comando spawn hasta que termine la acción.

```
expect {
password: {send "$pass\r"; exp_continue}
}
```

Y ya tendremos el primer script creado, solo debemos de guardar con CTRL+O y

salir con CTRL+X.

Le damos permisos de ejecución al script con **chmod +x enviarRespaldo.exp**

NOTA: el código fuente de este script esta al final del manual.

Script principal para hacer el respaldo

Para crear el script podemos usar el siguiente comando.

sudo nano hacerRespaldo.sh

Se nos abrirá un editor en la terminal, pondremos las primeras dos líneas siguientes:

#!/bin/bash
#vars

Estas indicaran el lenguaje en el que se escribirá el script, y que lo podremos ejecutar como un programa común.

Nota: Ir colocando las terminaciones de los if "IF", y tenerlo tabulado para una mejor comprensión lectora.

Establecemos las rutas de donde se guardarán las copias de las bases de datos-

backups_path="/var/lib/postgresql/12/respaldos/"

hay que asegurarse que la carpeta de respaldos este creada en esa ubicación, si no está creada la podemos hacer con el comando **mkdir respaldos.**

La siguiente variable será el nombre de nuestra base de datos que queremos respaldar, también es posible pasar el nombre como parámetro y obtenerlo por medio de las variables de bash como $1.

database=" seg_pac"

En las siguientes líneas se establecerá la fecha y la hora en la que será creado el respaldo

current_date_time=$(date +%F.%H-%M-%S)

En seguida haremos una consulta para guardar el numero de la unidad en un archivo temporal que se llama “id.txt”, el numero se separa de la línea completa que trae el archivo de texto.

psql -U postgres -h localhost -p 57361 -d seg_pac -c "SELECT id_unidad_archivo FROM catalogos.version_sistema;" > id.txt

numero=$(awk 'NR==3' id.txt)

Después eliminamos el archivo temporal “id.txt”.

rm id.txt

Creamos una nueva variable que se llamara “numeroId” aquí guardaremos el valor de la unidad separando los espacios que pueda tener en la variable anterior.

numeroId=$(echo $numero | tr -d '[[:space:]]')

y creamos el nombre del respaldo concatenando el número que acabamos de obtener.

```
backupNameId=$backups_path"bk"$numeroId
backupName=$backupNameId"_"$database
backupNameDate=$backupName"_"$current_date_time
echo $backupNameDate
```

Después podremos realizar el volcado de memoria para hacer el respaldo, es importante señalar que se requerirá la contraseña del usuario de postgres que sea propietario de la base de datos que quiere respaldar, a menos que se halla hecho el archivo “.**pgpass”** de pasos anteriores.

Además, utilizaremos el dump básico para crear el respaldo.

```
#dump
```

echo "Introduciendo la contraseña de usuario de postgres...."
echo "Creando dump de la base de datos...."
pg_dump -U postgres -h localhost -p 57361 -F c -d $database > $backupNameDate.sql &&

Note que al final del dump tiene "&&" esto es para que nos regrese una confirmación si se ha realizado la operación de forma exitosa.

Al realizar el dump de la base de datos, necesitamos comprimir el archivo para que su carga sea más liviana a la hora de subir al servidor y también para ahorrar espacio en el disco duro o memoria usb donde se almacene. La comprimimos y después borramos el archivo antiguo más pesado.

```
7z a $backupNameDate.7z $backupNameDate.sql
rm $backupNameDate.sql
```

Para comprobar que el volcado fue exitoso utilizaremos la siguiente línea:

```
if [ $? -eq 0 ];
then
echo "dump realizado con exito"
```

La variable "$?" guarda un "0" si tuvo éxito o un "1" si fallo.
Y así terminaríamos el script para realizar la copia de la base de datos en un directorio en el servidor.

Mandando la copia a un servidor externo

Si necesitamos enviar el respaldo a un servidor en concreto podemos realizarlo con el siguiente paso

echo "Revisando conexión a internet...."

podemos revisar la conexión a internet o al servidor de manera sencilla con un ping

ping slp-db.slpsalud.gob.mx -c 1 > tempPing.txt

en este ejemplo, se está haciendo ping a la dirección del servidor, con un conteo de 1 segundo para que el comando se detenga automáticamente, además de mandar a salida a un archivo de texto para que no se muestre en la terminal, y que será borrado después.

Comprobamos que la conexión fue exitosa

```
if [ $? -eq 0 ];
  then
```

Después de comprobarlo, será hora de usar el script que se creo que el punto anterior, para poder llamarlo colocamos la siguiente línea.

```
echo "Enviando el respaldo a el servidor.. ...."
   expect  /var/lib/postgresql/12/enviarRespaldo.exp
$backupNameDate.7z
```

Utilizando una memoria para guardar la copia

Continuando con el script que ya hemos hecho, podemos agregar la opción si deseamos guardar la copia en una memoria usb también.

Agregamos un “else” para el if del servidor.

Se listarán los dispositivos montados para poder determinar si la Usb está conectada y la salida del comando se escribirá en un archivo temporal que luego será eliminado

```
df -h > temp.txt
```

Buscaremos el directorio donde se montan los dispositivos

```
echo "Buscando dispositivos..."
echo "Respaldando en memoria Usb"
 dispositivos=$(grep -r "/dev/sdb1" temp.txt)
 echo $dispositivos
 echo "Creando temporal..."
```

Guardaremos el nombre de los dispositivos en la variable “dispositivos” gracias al filtro que aplicamos con la instrucción grep. Ahora comprobamos que la variable no esté vacía por lo que hacemos:

```
if [[ $dispositivos = "" ]];
then echo "No hay dispositivos conectados"
```

Si no está vacía continuamos con las siguientes instrucciones.

```
else
      IFS='/' read -r -a arr <<< $dispositivos
      cont=${#arr[@]}
      nomUsb=${arr[(($cont-1))]}
      user=${arr[(($cont-2))]}
```

```
echo "nombre de USB "$nomUsb
echo "nombre del Usuario "$user
```

Se lee la línea donde esta montado el dispositivo para así poder separar el nombre y del usuario que lo tiene montado.

Crearemos una especie de “try” ya que en bash no es posible una sentencia así, por lo que tenemos que poner la siguiente estructura, con ello nos aseguramos que la memoria este montada en el equipo y si no, procedemos a montarla manualmente en una ubicación que nos convenga.

```
if [ $? -eq  1 ];
then
        echo "No se encuentra montada"
        echo "Montando....."
{
              mount -t vfat /dev/sdb1 /media/usb && echo
              "Se ha montado correctamente" &&
        bandera=0
} ||
{
        echo "No se ha podido montar"
        bandera=1
}
        else
        echo "Ya está Montada"
        bandera=2
fi
```

La bandera que establecemos en la estructura anterior nos permitirá hacer la copia en la unidad montada manual o si ya se había montada automáticamente, ingresar el nombre de la memoria USB para que se copie en ella

```
echo "Borrando temporal....."
rm temp.txt
echo "Copiando a USB..."
if [ $bandera -eq 2 ];
        then
        cp $backupNameDate.7z /media/$user/$nomUsb &&
echo "Copia exitosa" || echo "Ha ocurrido un error"
elif [ $bandera -eq 0 ];
        then
        cp $(backupNameDate) /media/usb && echo "Copia
        exitosa" || echo "Ha ocurrido un error"
        echo "Desmontando"
        umount  /media/usb
```

```
else
        echo "Ha ocurrido un error...."
fi
```

Como parte final del script borramos los respaldos más antiguos y también el archivo temporal ping, para que no saturemos el disco duro.

```
echo "Borrando más antiguos..."
rm tempPing.txt
find /var/lib/postgresql/12/respaldos/ -mtime +1  -exec rm -f {} \;
echo "Saliendo..."
fi
```

Guardamos con CTRL+O y cerramos con CTRL +X.

Por ultimo crearemos un crontab del usuario root para que el script se ejecute cada cierto tiempo, que será cada medianoche.

```
crontab -e
00 00 * * * bash /var/lib/postgresql/12/hacerRespaldo.sh
```

Nota: al final del manual se encuentra el código fuente de este script

Aquí termina la instalación básica

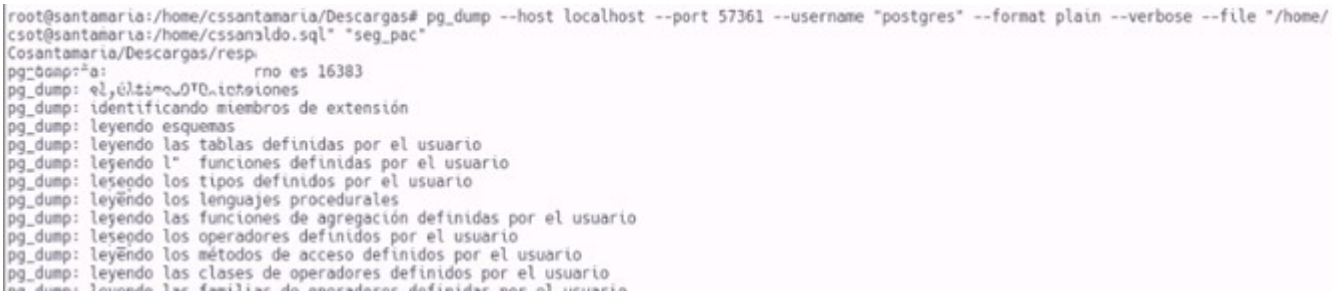
Configuraciones extras

Respaldar base de datos del Servidor por comando:

Si requiere un respaldo, se deberá de acceder al servidor fuente de datos y ejecutar los siguientes comandos:

Obtener la base de datos fuentes (respaldo).

pg_dump --host localhost --port 57361 --username "postgres" --format plain --verbose --file " [path\nombre_archivo.sql]" "seg_pac"



Nota: Este archivo comprimir y trasladarlo al servidor destino.

Servidor destino: Borrar base de datos en caso de que exista en el servidor.

psql -h localhost -U postgres -p 57361 -c 'DROP DATABASE "seg_pac";'

Crea base de datos:
psql -h localhost -U postgres -p 57361 -c 'CREATE DATABASE "seg_pac";'
psql -h localhost -U sin_triggers_ -p 57361 seg_pac < [path\nombre_archivo.sql]

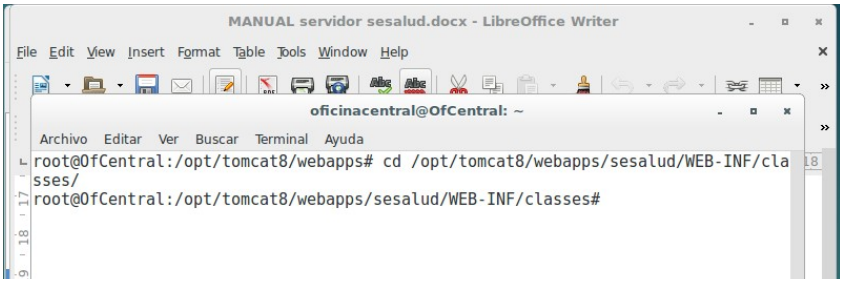
El SeSalud acceda a otra base de datos:

La aplicación distribuible de SeSalud es un archivo .war empaquetado previamente, el cual contiene el proyecto listo para desplegar en el ambiente Tomcat y en el manual de instalación de aplicaciones vimos como instalarlo por ello continuamos con la parte de configuración.

Una vez descomprimido se debe modificar el archivo de propiedades que contiene la dirección del servidor y el nombre de la base de datos, puede ser hecho de la

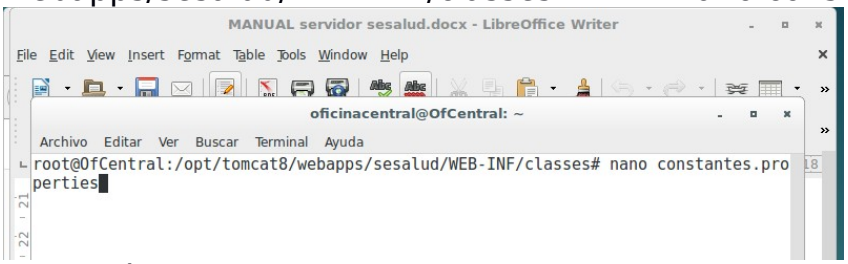
siguiente manera:

- # cd /opt/tomcat8/webapps/sesalud/WEB-INF/classes/



Ejecución "**cd /opt/tomcat8/webapps/sesalud/WEB-INF/classes/**"

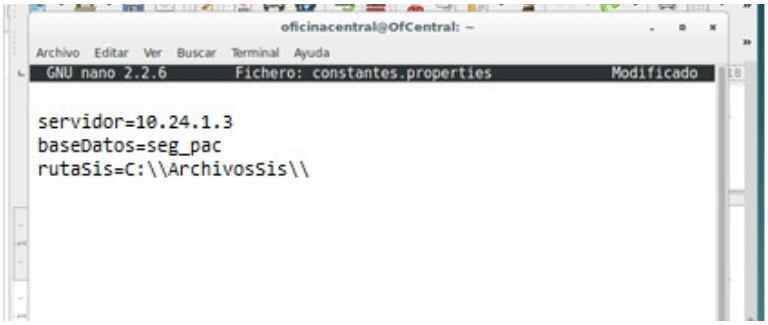
- webapps/sesalud/WEB-INF/classes# nano constantes.properties.



Ejecución comando "nano constantes.properties"

Y se abrirá el archivo de texto en el cual deberemos personalizar la dirección IP del servidor y el nombre de la Base de Datos siguiendo las normas que tiene actualmente, es decir, texto sin comillas.

servidor=localhost
baseDatos=seg_pac
rutaSis=C:\\ArchivosSis\\



Configuración archivo.

Una vez hecho esto podremos entrar desde el navegador mediante la dirección IP de nuestro servidor y el respectivo puerto, por ejemplo: http://192.168.1.250:8080/sesalud y acceder con el usuario administrador del sistema. En este caso usuario: **adminh** contraseña: **adminh.**

Códigos Fuente

enviarRespaldo.exp

```
#!/usr/bin/expect
set nombre [lindex $argv 0];
```

```
set timeout -1
spawn scp $nombre  sesalud@slp-db.slpsalud.gob.mx:/home/sesalud/respaldos
set pass sesalud
expect {
password: {send "$pass\r"; exp_continue}
}
```

hacerRespaldo.sh

```
#!/bin/bash
#vars

backups_path="/var/lib/postgresql/12/respaldos/"
database="seg_pac"
current_date_time=$(date +%F.%H-%M-%S)
psql -U postgres -h localhost -p 57361 -d seg_pac -c "SELECT id_unidad_archivo
FROM catalogos.version_sistema;" > id.txt
numero=$(awk 'NR==3' id.txt)
rm id.txt
numeroId=$(echo $numero | tr -d '[[:space:]]')
backupNameId=$backups_path"bk"$numeroId
backupName=$backupNameId"_"$database
backupNameDate=$backupName"_"$current_date_time
echo $backupNameDate
#dump
echo "Introduciendo la contraseña de usuario de postgres...."
echo "Creando dump de la base de datos...."
pg_dump -U postgres -h localhost -p 57361 -F c -d $database >
$backupNameDate.sql &&
7z a $backupNameDate.7z $backupNameDate.sql
rm $backupNameDate.sql
#Verifica que la contraseña de postgres sea la correcta
if [ $? -eq 0 ];
then
    echo "dump realizado con exito"
    echo "Revisando conexion a internet...."
    #revisando la conexion a internet con el comando ping a la dirección 8.8.8.8
    ping slp-db.slpsalud.gob.mx -c 1 > tempPing.txt

    if [ $? -eq 0 ];
    then
        echo "Envando el respaldo a el servidor.. ...."
        expect /var/lib/postgresql/12/enviarRespaldo.exp $backupNameDate.7z
    else
        echo "No se pudo conectar al servidor.."
        echo "Respaldando en memoria Usb"
```

```
#Listaremos los dispositivos montados para poder determinar si la Usb esta
conectada
#la salida se escribira en un archivo que luego sera eliminado
df -h > temp.txt
#buscaremos el directorio donde se montan los dispositivos
echo "Buscando dispositivos...."
dispositivos=$(grep -r "/dev/sdb1" temp.txt)
echo $dispositivos
echo "Creando temporal..."

if [[ $dispositivos = "" ]];
then
   echo "No hay dispositivos conectados"
else
#se lee la linea que tiene montado el dispositivo para
#saber su nombre y la carpeta donde esta montado
   IFS="/" read -r -a arr <<< "$dispositivos"
   cont=${#arr[@]}
   nomUsb=${arr[(($cont-1))]}
   user=${arr[(($cont-2))]}
   echo "nombre de USB "$nomUsb
   echo "nombre del Usuario "$user
   if [ $? -eq  1 ];
   then
      echo "No se encuentra montada"
      echo "Montando....."
      {
         mount -t vfat /dev/sdb1 /media/usb && echo "Se ha montado
correctamente" &&
         bandera=0
      } ||
      {
         echo "No montada"
         bandera=1
      }
   else
      echo "Montada"
      bandera=2
   fi

   echo "Borrando temporal....."
   rm temp.txt
   echo "Copiando a USB..."

   if [ $bandera -eq 2 ];
   then
```

```
            cp $backupNameDate.7z /media/$user/$nomUsb && echo "Copia
exitosa" || echo "Ha ocurrido un error"
        elif [ $bandera -eq 0 ];
        then
            cp $(backupNameDate) /media/usb && echo "Copia exitosa" || echo
"Ha ocurrido un error"
            echo "Desmontando"
            umount  /media/usb
        else
            echo "Ha ocurrido un error...."
        fi
    fi
fi

    echo "Borrando mas antiguos..."
    rm tempPing.txt
    find /var/lib/postgresql/12/respaldos/ -mtime +1  -exec rm -f {} \;
    echo "Saliendo...."
fi
```

links:
https://blog.desdelinux.net/habilitar-el-usuario-root-en-ubuntu/
https://ubunlog.com/tomcat-9-instalacion-ubuntu-18-04/
https://miracomosehace.com/instalar-escritorio-remoto-anydesk-linux-ubuntu-consola/
https://www.it-swarm-es.com/es/postgresql/pgadmin-el-paquete-pgadmin4-no-tiene-candidato-de-instalacion/811882615/
https://geeksencuarentena.com/linux/como-instalar-postgresql-y-pgadmin4-en-ubuntu-20-04/
https://wiki.postgresql.org/wiki/Apt
https://www.postgresql.org/download/linux/ubuntu/

https://linuxhint.com/install-pgadmin4-ubuntu/